**Design**

Breakdown of problem

To design the system efficiently I must plan the basic functions needed. I analyse all the features necessary and then when creating the system, I will then progress upon these basic needs. Doing this will make sure that I get all the basic functions ready.

The first, most important feature I need is the ability to login and determine the type of account. This login must be smooth and quick yet manage to be 100% effective and correct. This is the most important feature as it is the only way to access the system without this the rest of the system will not be able to function.

An unnecessary feature that I intend to include is a map. This doesn’t affect the running of the system however it will improve the usability of this new system. A map means that people can check to see if lockers are available where they desire before going to check.

When logged in I need to make the system display to the user which lockers, in the unit, are either occupied or vacant. Building on this they then need to be able to select and open only the vacant lockers. On the contrary if the user is an admin they need a feature specific to them that overrides this rule allowing them to open any locker of their choice.

After a locker has been booked someone must be able to see how long they have left on the locker. This means that they can check if they can use it during a lesson or not. This information is important as if they go over the limit they have then it will be noted, potentially incurring a fine. This is important to the college as they must be able to see if people are abusing their system or not.

A user must have a feature available to them to be able to end their booking early. This is important as lockers free up faster which means that more people would be able to use lockers, making the system an overall success. This is a crucial feature as it could decide whether the improvement has successfully caused lockers to be a more used system about the college.

The final required feature in the system is the ability to logout. When a user has finished completing all their work they will then want to leave the system. A logout is essential as without it someone will be able to compromise security, allowing them access to whosevers account was previously logged in. For a logout to be successful all confidential information of the previous user must be erased or hidden from view of the front end at a minimum.

After I ensure that all the basic parts of the system are functional I must the think about how the program must cater to everyone in the college. Ensuring that as many people as possible adapt and successfully use the new system. This is the only way that the system will be feasible, it must prove in the pilot changeover that people will use it. In order to do this, I must plan to allow for people of all disabilities to use this system.

The QR scanning system must be simply accessed, hence I must put it at a height that ensures that everyone can see and use it. A reasonable height would be half way down the locker unit. I believe that all people would then be able to comfortably scan their card and use the touch screen.

To ensure that people who have lost their card or forgotten it I must be sure that the temporary stickers given out are also functional with the new system. As a backup to this admins and staff members will be able to open lockers of students. Doing this means that even if a pupil forgot their card the teacher they currently have could open the locker if the lesson so requires.

From my discussion, a valuable point was given, people who are colour blind may not be able to tell the difference between the green (available lockers) and red (in use lockers). This means that I must guarantee they can still use this system by not making the display colour orientated, there must also be a text aspect. Doing this not only caters for colour blindness but also dyslexia. This is done by simply having the original problem catering for colour blindness. By having the colour scheme, it means people with reading disabilities aren’t required to read to be able to understand the system.

Flow chart

This flow chart has been designed to give a basic view of all the key elements to the system. It shows how the screens link to each other and also the basic actions that happen in the back end. This means that I am able to constantly analyse my project looking at the flaws of it and then remember this for the future of the project. As well as this the simplicity

I used dead end connectors to keep the image clean. When a dead is found it continues at the similar connector but coloured in. This has kept my design clean as several different screens do link to one screen.

Individual screen designs

After planning out the features that are required for the system to function, I then had to design how I would display this as a front end. The front end is the only part of the system that is visible to the user so it should be informative and have no confusion in operation. Each design is only a design so the final program or prototypes may drop specific features or even add in more. This is to assist with creating of the program and give a reference to what the system was pictured as.















Screen design analysis

Home screen

The home screen is designed to be simplistic. It is going to provide gateways to the next section desired by the user whether they are an admin or not. I have only kept very basic features in the design trying to keep only the functions that are necessary, doing this will allow for the logging in process to be quick and understandable. This is the key concept to the upgrade system so I will be focusing on this in every screen. Currently in the top left there is a label that says “clock” if possible I will code my own clock that will run in the background on a separate thread. A clock isn’t needed however it will be a nice addition for the user.

Student menu

The user GUI is a screen primarily designed to give a clear representation of everything that the students can access. It will show them what they can access and also provide them with a method of accessing via a meaningful identifier. In these designs, I have planned to keep the layout similar from the home screen to this menu. I think of this as a good idea as it means the user doesn’t have to go searching around the screen, similar objects will be in similar places across multiple screens.

Admin menu

The admin menu has a very similar purpose to the student menu. The only difference is that they provide access to different parts of the system and one requires more privileges to access then the other. On the admin page, the logout button is crucial, it is the most important button of the entire system. If this button fails, then the admin details may remain on the system which could cause a breach in the security of the system. This button must be the only way that the admin can leave the system meaning that if their account is accessed without permission then it was not the fault of the system.

Map

I decided that a map will be a good idea as it will provide useful information to anyone who is using the system. Placing it on a new screen was required as not all people would want to view it. It also allows another method to be ran when changing to the map screen.

Button page

The button page has one purpose, to select a locker to open and then with correct validation open that locker. This means that the page should be simplistic, when the user clicks on the screen they should already know all they need to know about what will be on the screen and what they then have to do to fulfil their required actions. The button page will display an array of buttons all linking to a locker, except one which is a return button used to exit the program. The buttons have been designed to ensure ease of access is most important; to ensure this is done they give both text and colour visuals to provide information of if they are in use or not.

Message prompts

Throughout the system there will be a series of confirmation messages. These will be informative and they are designed to be quick and easy. Each message will differ slightly but the main concept that they will only contain a brief message and a button will be present on all of them. The messages only occur when the user has selected something that they cannot do or they have successfully performed an action. On both screens there is no data to be displayed only a sentence this means that the user won’t spend long reading it and the time consumption will be minimal.

Inputs of the system

There are going to be complex inputs in the system and tabulating them means that they are easy to keep track off. The table means that when it comes to designing and creating the final code I will be able to check off each input and make sure that all of them are in the final project.

|  |  |  |
| --- | --- | --- |
| Input name | Input data | Description |
| Persons database. |  | |
| -Person’s name and ID. | The name and ID from the database of the user. | A name and ID will be read that links to the QR code the user shows. This gives a verification aspect as the user has to enter that they are the person on the screen. |
| -Person’s admin status. | A Boolean value if the user is regarded as an admin or not. | From this value, the user gets given certain access to parts of the system. |
| -Person’s locker usage. | A value of how many lockers the user has. | This allows the system to check if the user is allowed to use more lockers. |
| -Person’s locker time remaining | A time out on the user’s locker. | This value will allow to see if a user’s locker is overdue and they are abusing their time given. |
| -Person’s Runshaw pay status (potentially) | A numeric value of how much money is on the users account. | The user may only be able to have a set number of lockers over the year. So, the system may be able to deny access. |
| Unit status | Inputs whether the locker unit is powered on or off. | An admin has the ability to turn on and off a unit. A locker unit must be able to know what this Boolean value is or they could just give access even when they are meant to be turned off. |
| Map details | The current amount of free lockers on each of the units around college. | Storing all of the unit’s status’ on the one program would be inefficient. I am going to store the values on a database. When someone goes to access the map page data will then be read in from this database to give the numeric values required. |
| Clock | A time value. | To provide an accurate time value I will use multi-threading to constantly input the time and then update the clock at a set time. |
| Time remaining | A value of when the locker usage is due to be up | I am planning on creating a database that will show the values of students with overdue locker usage. From this staff members will be able to quickly access this and assign fines if necessary. |

Files and data structures

Databases

In the project, I will use a variety of files, each of these files will be accessed as a database. They will contain information that is linked to a specific subject for example, the data that is linked to a student, consisting of their id name and status. To keep the code secure and efficient each object will have a single text file linked to it, meaning there will be more files which does increase the storage but for a good reason, making the system more durable.

Users (database)

The user’s database is going to contain the records of everyone that can use the system. This means that within each record there must be a unique primary key field which will identify whose record is who’s. I have decided that the best field will be the ID of each student and if I decide to sort this database I will sort it by ascending order in ID’s. Each record will also contain fields linking each person to the locker system. These items will be about the lockers and when they login I will bring through all these values and then nullify them when the user logs out.

Unit usage (database)

This file is going to store the number of lockers available in each unit, it will also store whether they are on or off in a Boolean value. The only times when this file is going is when someone wishes to access the map page and when someone uses a locker. If the user decides to view the map page then a procedure will run and will get back the current information on the locker usage across all units. This will provide a near enough live feed for people accessing the map feature.

Overdue (database)

The overdue database is going to be a way of the staff members analysing those who are abusing the locker system. When someone goes to end their locker time it will analyse if they have run over the time limit, from this the system will then output to the text file their basic credentials and how long they were overdue. The college can then get this text file and quickly print or view it and carry out whatever they feel is necessary.

Units

The different units around college will have to output to various parts of the usage database, to do this each individual unit will run a slightly different program, which uses the same classes. The main parts of the code will be repeated, the one small difference will be a value that determines where it outputs as to how many lockers are in use.

Methods of access

In this project, I am going to get the major amount of data required from databases in the form of text files. To do this the program is going to read and write to several separate text files. The text files will have meaningful identifiers related to what was stated before in the files and data structures section - databases. Currently I am planning on having three files that are going to store vital data. Within the text files are will use a special character in order to separate different pieces of information. For example, the record related to a student will have many fields in and I am required to access these apart from each other. Ideally, I will encrypt all of the data but this may not be achievable in the time frame given.

|  |  |  |  |
| --- | --- | --- | --- |
| Filename | Approximated records | Values stored | Example |
| Users.txt | >1000 | This will store all the people within the college that are able to use lockers, including: Name, ID, Admin Status, Current locker status, Runshaw pay amount | MaxStevenson|STE15155284|0|0|100 |
| Usage.txt | 20 | A title of all the locker units across the campus including a Boolean value if they are turned on and how many lockers they have available. If a unit is turned off then it will register zero lockers available | Rydal|1|10  Griezdale|1|7  Dalehead|0|0 |
| Overdue.txt | <50 | Storing the name and ID of who is using the locker, which locker it is and how far overdue it is valued in minutes | MaxStevenson |STE15155284 |Rydal02 |60 |

Validation/ Verification

A verification technique will be used in the system. Once the user scans in their basic credentials will be displayed then they must click enter. If these are not their details, then they wouldn’t enter this is a form of human verification. It doesn’t have a technical management but it should reduce human error.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Field | Validation type | Validation rule | Explanation |
| 1 | QR | Type | Must be a QR code. | For the system to function it must only activate if a scanner detects a QR code. |
| Lookup | Must be in database. | Once the system is sure the input is QR it must then check if it is valid and an item in a database. |
| 2 | Login details | Type | Must be either string or integer | Checking each input from a text file to ensure that each detail is the correct type. |
| 3 | Admin value | Lookup | Validating whether the user has admin privileges. | If someone wants to access the admin screen then first of all they must validate their user level. |
| 4 | Locker unlocking | Lookup | The locker must be free | When someone wishes to unlock a locker, the system will validate if that locker is free or not in accordance to an array. This doesn’t apply when admins are opening a locker |
| 5 | Locker usage | Length | Must be less than a set amount | The lockers can only be rented out for a certain amount of time so only specific amount of time can be used. |
| 6 | Overdue | Range | Less than 0 | This is to validate that the locker usage is actually overdue, returning a negative value. If the value is positive then it means the locker was used under their limit. |

Processing stages

Process

|  |  |  |
| --- | --- | --- |
| No. | Process | Relations |
| 1 | Scanning QR code | None |
| 2 | Checking QR code to be valid | Users database |
| 3 | Displaying login details |  |
| 4 | Access student home page | Data already gathered from (2) |
| 5 | Check current status | Users database linked via data already gathered from (2) |
| 6 | Open a locker | Usage database |
| 6.1 | If overdue note in file | Users database + Overdue database |
| 7 | Accessing map page | Usage database |
| 8 | Access admin home page | Data already gathered from (2) |
| 9 | Turn on / off system | Usage database |
| 10 | Logout | None |

Pseudo code-

For my pseudo code, I have decided to write up the basic core functions that will be used in my code. This assist as it means that I can simply read my code and other people are able to read it even if they do not have a concept of the language that I am using. This means that I can get an outside opinion without them thinking about how the system will be coded; giving a thought as to how this system will function over is it possible to code.

**Start**

**Declare panels**

**Declare page variables**

**Declare Tab**

**Declate generic varialbes**

**End**

**Start // start GUI**

**Declare grid**

**Call pagescreations**

**// Add created pages to main tab**

**End**

**Start // buttonupdate**

**IF (lockerAvailability[0] == 0)**

**THEN**

**SET theButtons[0][0] background Green**

**END IF**

**IF (lockerAvailability[0] == 1)**

**THEN**

**SET theButtons[0][0] background Red**

**END IF**

**// Do SELECTION for all other lockers**

**End**

**Start // mapUpdate**

**DECLARE count = 0**

**OPEN FILE usage.txt for INPUT**

**WHILE(count < usage.txt.length())**

**SET usage readline // Use buffered reader**

**SET textfield(usage)**

**END WHILE**

**CLOSE FILE**

**End**

**Start //Gather user details**

**INPUT currentID //Get from QR code**

**DECLARE count = 0**

**DECLARE temp**

**OPEN FILE users.txt FOR INPUT**

**WHILE(count < users.txt.length())**

**SET temp readline //Use BufferedReader from java**

**IF (temp = currentID)**

**THEN**

**SET ID, Name, Admin Status, Current Locker status, Runshaw pay amount**

**END IF**

**END WHILE**

**CLOSE FILE**

**OUTPUT (Name + ID)**

**End**

**Start //Admin page access**

**INPUT buttonClick //Action listners**

**IF (AdminStatus == 1)**

**THEN**

**SetSelectedIndex(Admin page)**

**END IF**

**IF (AdminStatus == 0)**

**THEN**

**OUTPUT (Sorry not admin)**

**END IF**

**End**

**Start //Student page access**

**INPUT buttponClick //Action listners**

**SetSelectedIndex(Student page)**

**END**

**Start //Map page access**

**INPUT buttonClick //Action listners**

**CALL mapUpdate()**

**SetSelectedIndex(Map page)**

**End**

**Start //Display status**

**INPUT ButtonClick //Action listners**

**OUTPUT (Current Locker status)**

**End**

**Start //Open locker**

**INPUT ButtonClick //Specific locker they chose**

**OUTPUT (Locker has been unlocked)**

**Open locker**

**LockerEnd = getTime() //Takes current time**

**SetSelectedIndex(Previous page)**

**IF(LockerEnd > LockerStart + Booking)**

**THEN**

**OPEN FILE overdue.txt FOR OUTPUT**

**OUTPUT (ID,NAME,|LockerEnd - LockerStart - Booking|)**

**CLOSE FILE**

**END IF**

**End**

**Start // Turn off/on system**

**INPUT buttonClick //Action listners**

**OPEN Usage.txt for INPUT**

**SET temp READ line**

**DISSOCIATE temp**

**IF (buttonClick == turnOn)**

**THEN**

**OPEN Usage.txt for OUTPUT**

**WRITE to Usage.txt Rydal|1|10**

**END IF**

**IF (buttonClick == turnOff)**

**THEN**

**OPEN Usage.txt for OUTPUT**

**WRITE to Usage.txt Rydal|0|10**

**END IF**

**CLOSE FILE**

**End**

**Start main**

**Call start gui**

**End**